Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Pato Branco
Disciplina: Estruturas de Dados, Pesquisa e Ordenação
Professora: Emanoeli Madalosso
Curso: Tecnologia em Análise e Desenvolvimento de Sistemas

Avaliação 1 - 20/09/2019

1) (4,0 pontos) Para este programa você deve criar dois vetores do mesmo tamanho (os itens dos vetores podem ser lidos pelo teclado, gerados aleatoriamente, etc). Implemente as funções solicitadas abaixo. Você deve usar ponteiros para percorrer/manipular os itens dos vetores. Obs.: Nesta questão você pode criar as funções no mesmo arquivo se preferir.

a) (2,0 pontos) Escreva uma função que recebe dois vetores e faz a troca dos elementos do primeiro vetor com os elementos do segundo vetor. Por exemplo:

```
Vetor 1 antes da troca:
1     2     3     4     5
Vetor 2 antes da troca:
10    20    30    40    50

Vetor 1 após a troca:
10    20    30    40    50
Vetor 2 após a troca:
1     2     3     4     5
```

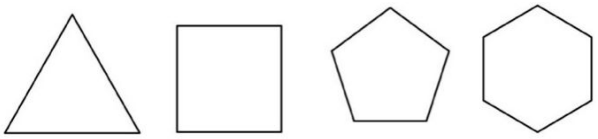
b) (2,0 pontos) Escreva uma função que recebe dois vetores e retorna um terceiro vetor contendo a soma dos dois vetores, item a item. Por exemplo:

```
Vetor 1:
1     2     3     4     5
Vetor 2:
10    20    30    40    50
Vetor 3:
11    22    33    44    55
```

2) (5,0 pontos) A estrutura abaixo representa um ponto do plano 2D:

```
typedef struct ponto Ponto;
struct ponto {
      float x;
      float y;
};
```

Sabemos que um polígono é uma figura fechada formada por segmentos de retas, portanto, deve ser formada por ao menos 3 pontos. As figuras abaixo são exemplos de polígonos:



Considerando a estrutura ponto definida acima, um polígono pode ser representado por um vetor de pontos:

- O polígono do tipo triângulo pode ser representado por um vetor de 3 pontos;
- O polígono do tipo quadrado pode ser representado por um vetor de 4 pontos;

E assim por diante. Desta maneira, crie um Tipo Abstrato de Dados “Polígono”, baseado na estrutura “Ponto” que foi fornecida. O TAD deve conter as funções listadas abaixo. Você pode criar funções adicionais se achar necessário.

a) **(0,6 ponto) aloca_memoria_poligono**: recebe a quantidade de pontos que formam o polígono, alocando memória para o vetor de pontos que irá representá-lo. Deve retornar o ponteiro para o primeiro endereço alocado;
b) **(0,6 ponto) libera_memoria_poligono**: libera a memória ocupada pelo vetor de pontos do polígono;
c) **(0,6 ponto) cadastra_ponto**: cadastra um ponto do polígono;
d) **(0,6 ponto) imprime_pontos_poligono**: recebe o vetor de pontos do polígono e imprime as coordenadas de cada ponto;
e) **(0,8 ponto) calcula_distancia_entre_dois_pontos**: recebe dois pontos do polígono e retorna a distância entre eles (a distância entre dois pontos forma um lado do polígono). Obs.: pode usar as funções da math.h;
$$distância = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$
f) **(0,9 ponto) calcula_lados_poligono**: recebe o vetor de pontos do polígono e retorna um vetor de *float* contendo as medidas dos lados do polígono: para cada ponto, calcule a distância entre ele e o próximo ponto. Ao final, lembre-se de fechar o polígono calculando a medida do lado formado pelo último ponto e pelo primeiro ponto, incluindo este lado no vetor.
g) **(0,9 ponto) calcula_perimetro_poligono**: recebe o vetor de pontos do polígono, calcula e retorna o perímetro do polígono (a soma de todos os seus lados).

Na função main testar as funções: alocar memória para o vetor de pontos do polígono, cadastrar os pontos do polígono, imprimir o polígono cadastrado, calcular e imprimir o perímetro do polígono e ao final liberar a memória que foi alocada. Exemplos para teste:

```
Polígono 1:
Ponto 1: 0, 0
Ponto 2: 10, 0
Ponto 3: 5, 5
Primétro: 21.14
```

```
Polígono 2:
Ponto 1: 0, 5
Ponto 2: 0, 0
Ponto 3: 10, 0
Ponto 4: 10, 5
Perímetro: 30.00
```

```
Polígono 3:
Ponto 1: 0, 16
Ponto 2: 0, 8
Ponto 3: 2.5, 0
Ponto 4: 12.5, 0
Ponto 5: 11, 8
Ponto 6: 5, 8
Ponto 7: 5, 16
Perímetro: 53.52
```